8.1 DATA REPRESENTATION AND STORAGE

This section describes the bit and byte numbering conventions used in this document, and the storage methods for integers and floating point numbers. This information is especially critical when transporting data from one computer architecture to another. Without special handling, data produced on one system may be unusable on another due to differences in internal data storage.

Within this section, a byte is defined as containing 8 bits (i.e., an octet), and a word can be either 16 or 32 bits in length. In all cases, the least significant bit (lsb) is designated as bit 0 and has a base10 value of $2^0 = 1$. Therefore, in a 32-bit word the most significant bit (msb) is designated as bit 31, and has a base10 value of $2^{31} = 2,147,483,648$.

For signed binary integers, the msb represents the sign of the number. The remaining bits (bits 14 through 0 for 16-bit words and 30 through 0 for 32-bit words) are used to designate the magnitude of the number. Positive binary integers are in true binary notation with the sign bit set to zero. Negative binary integers are in two's-complement notation with sign bit set to one. Negative binary integers are formed in two's-complement notation by inverting each bit of the positive binary integer and adding one.

Unsigned binary integers use all bits including the msb to represent the magnitude of the number. Unfortunately, this data type is not supported by all computer languages (e.g., FORTRAN), so additional data manipulation may be necessary. In the case of reading a 16-bit unsigned integer (DATA), a FORTRAN user could use the following pseudocode example to extract the actual value (VALUE):

```
...
INTEGER*2 DATA
INTEGER*4 VALUE
...
READ DATA
IF (DATA .LT. 0) THEN
     VALUE = 65536 + DATA
ELSE
     VALUE = DATA
ENDIF
...
```

To provide maximum portability of the data sets across different computer platforms, floating point data is represented by scaled integers. Scaled integers can be either signed or unsigned, and are simply floating point numbers multiplied by a fixed scaling factor until sufficient accuracy can be stored in integer form. For example, the floating point value 1.23 can be multiplied by 10^2 to achieve an integer value of 123. In the NOAA Level 1b data sets, the scaling factors are powers of ten, and only the exponents (2 in the previous example) are documented within the data set. To recover the original floating point value, divide the integer value by ten raised to the given exponent.

A major problem impeding the free transport of binary data from one computer system to another is the "Big Endian - Little Endian" dichotomy. "Big Endian" systems (e.g. IBM 370, Macintosh, SGI, Sun SPARC) store bytes of binary data in reverse order relative to "Little Endian" systems (e.g. IBM PC, DEC Alpha). For example, a 32-bit hexadecimal value of x01020304 written to a binary file by a Big Endian system would be read from the file as x04030201 by a Little Endian system. NOAA Level 1b data sets are in Big Endian order; users with Little Endian systems must include an additional byte-swapping step when reading binary data from NOAA Level 1b data sets. Some processors support byte swapping in their instruction sets, but others must use compiler dependent functions.